项目 1　了解 ASP.NET 与 Visual Studio 2012

学习任务

本项目通过使用 C#语言 ASP.NET 来开发 Web 应用程序，在此过程中通过实践来熟悉 ASP.NET 和 Visual Studio 2012 的集成开发环境（IDE），理解并掌握 ASP.NET Web 应用程序的构成。

学习目标

◎ 熟悉 ASP.NET 2.0 的集成开发环境
◎ 学会在 Visual Studio 2012 中新建 Web 应用程序
◎ 掌握 ASP.NET 程序构成与处理过程

1.1　设计“Hello VS2012”

学习要点

1. 掌握新建 Web 应用程序
2. 熟悉 ASP.NET 的集成开发环境（IDE）
3. 熟悉 ASP.NET 的程序运行环境

导学实践，跟我学

【案例】设计“ Hello VS2012 ”。

Visual Studio 2012 是微软推出的一个基于 Windows 平台开发的集成开发环境。在此将以一个简单的“ Hello VS2012 ”为例讲解在 Visual Studio 2012 中开发 ASP.NET Web 应用程序的基本过程。

具体步骤如下。

（1）打开 Visual Studio2012 后，执行“ 文件 ”→“ 新建网站 ”命令，在弹出的“ 新建网站 ”对话框中，语言选择“ Visual C# ”，类型选择“ ASP.NET 空网站 ”，并单击“ 确定 ”按钮，如图 1-1 所示。

小提示

在“位置”中有三个选项：文件系统、HTTP、FTP。文件系统表示选择此种方式由系统建立一个临时 HTTP 服务器，端口随机生成，外部不能访问，安全性高；HTTP 表示如果本地计算机架设了 HTTP 服务器，可以直接将文件放在一个配置好的 Web 目录中；FTP 表示将文件存放在远程目录中，这适合对已经存在的 Web 应用程序做小方面修改。

（2）在 WebHello 的网站开发环境中，可以在“ 解决方案资源管理器 ”看到一个“ Web.Config ”文件并右击该文件，在弹出的快捷菜单中选择“ 添加 ”→“ 添加新项 ”选项，如图 1-2 所示。

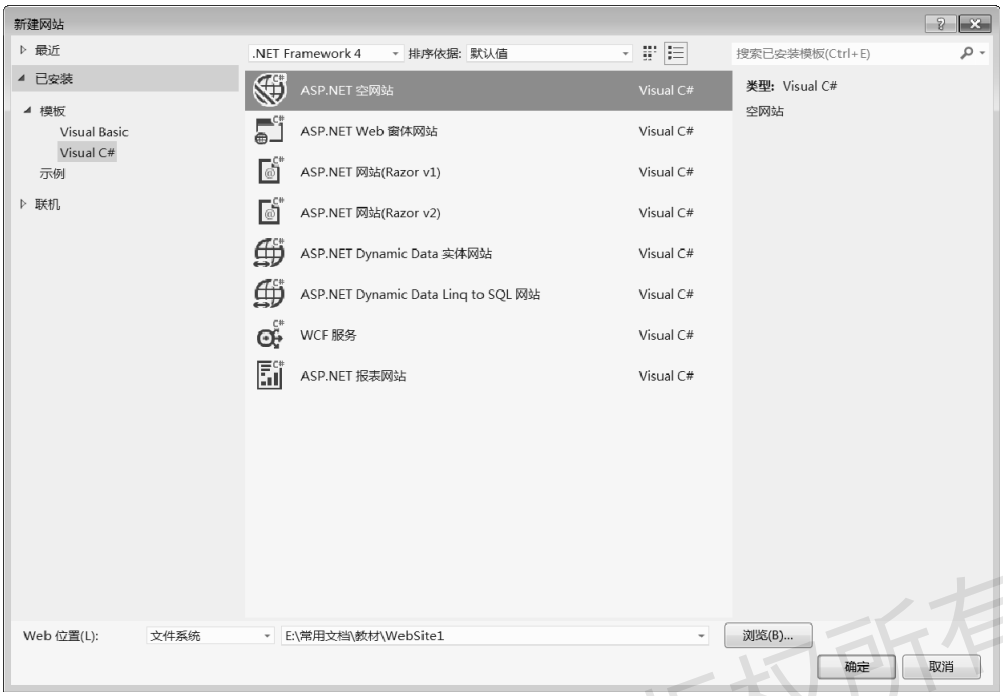

图 1-1　新建网站

图 1-2　添加 Web 窗体

在弹出的窗口中选择“ Web 窗体 ”选项，窗体名称为“ Default.aspx ”，然后单击“ 添加 ”按钮，便在网站的目录下创建了“ Default.aspx ”页面，如图 1-3 所示。

小提示

Default.aspx 和 Default.aspx.cs 分别对应页面设计和后台代码，可以这样理解：Default.aspx 存放的是设计，Default.aspx.cs 存放的是隐藏的代码页。这在一定程度上实现了代码与页面分离。

(3) 在图中单击“ 设计 ”标签 (左下角有标注), 就可以转到“ 设计 ”视图, 在工具箱里将“ label ”控件拖入到“ 设计 ”视图中，并命名为“ lblHello ”，如图 1-4 所示。

(4) 在“ 设计视图 ”下，双击空白位置，会从 Default.aspx 页面转到 Default.aspx.cs 页面，在页面载入事件 Page_Load()中输入如图 1-5 所示的代码。

代码如下：

```
    this.lblHello.Text = "Hello VS2012";
```

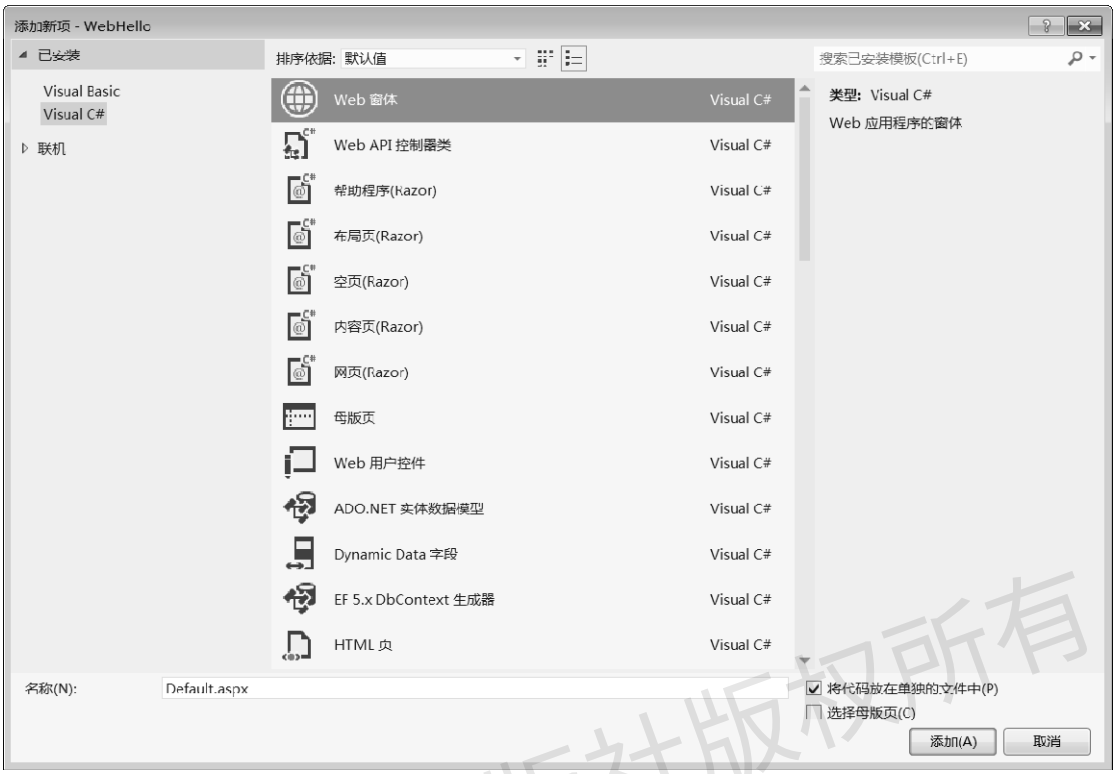

图 1-3　新建 Web 窗体

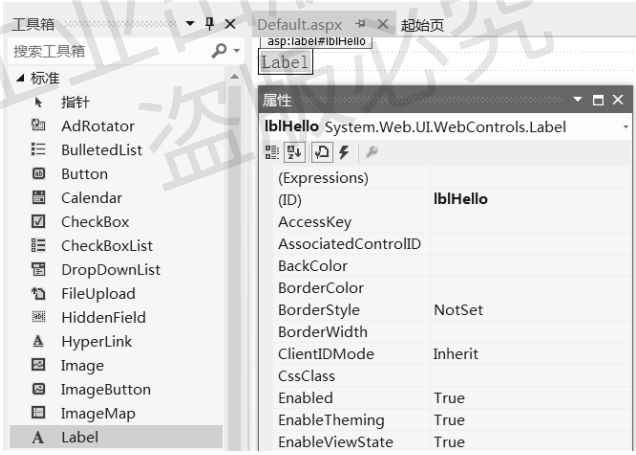

图 1-4　设置 Label 属性

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Web;
5  using System.Web.UI;
6  using System.Web.UI.WebControls;
7
8  public partial class _Default : System.Web.UI.Page
9  {
10     protected void Page_Load(object sender, EventArgs e)
11     {
12         this.lblHello.Text = "Hello VS2012";
13     }
14 }
```

图 1-5　后台代码

（5）执行“调试”→“启动调试”或“开始执行（不调试）”命令，将出现如图 1-6 所示的效果。

图 1-6　程序运行效果

示例解释

（1）一般来说，第一次运行网站时会弹出“未启用调试”。在“未启用调试”中，用户可以选择添加 Web.config，单击“确定”按钮，就可以将 Web.config 文件添加到 Web 应用程序中，并启动调试功能。

（2）本案例利用控件 Label 实现了一个简单的“Hello VS2012”的 Web 应用程序（网站），在.NET 框架下利用 C#语言实现了基于 B/S 架构的网站。用 C#语言开发的网站代码页*.aspx.cs。很多时候，可以这样理解：.cs 文件内存放后台操作代码，而.aspx 文件只是存放各个控件的代码，处理程序代码一般放在.cs 文件中。

背景知识

1．Visual Studio 2012 简介

Visual Studio 2012 是微软推出的一款功能完整的开发环境，是 Windows 平台最流行的应用程序开发平台。软件采用了新版 Windows 的核心功能，因此 Visual Studio 2012 只能在系统为 Windows 7 或更高版本的计算机上才能运行。

Visual Studio 2012 整个界面经过了重新设计，简化了工作流程，并且提供了访问常用工具的捷径。工具栏经过了简化，减少了选项卡的混乱性，用户可以使用全新快速的方式找到代码。所有这些改变都可以让用户更轻松地导航应用程序，以用户喜爱的方式工作。

随着 Windows 8 的发布，Visual Studio 2012 提供了新的模板、设计工具及测试和调试工具——在尽可能短的时间内构建具有强大吸引力的应用程序所需要的一切。

对于 Web 开发，Visual Studio 2012 也为用户提供了新的模板、更优秀的发布工具和对新标准（如 HTML5 和 CSS3）的全面支持，以及 ASP.NET 中的最新优势。此外，用户还可以利用 Page Inspector 在 IDE 中与正在编码的页面进行交互，从而更轻松地进行调试。那么对于移动设备又如何呢？有了 ASP.NET，用户便可以使用优化的控件针对手机、平板电脑及其他小屏幕来创建应用程序。

2．Visual Studio 2012 界面介绍

在 Visual Studio 2012 界面整体分为菜单工具栏、工具箱、代码视图编辑区、解决方案管理区和错误调试显示窗口，如图 1-7 所示。

图 1-7　Visual Studio 2012 界面

菜单工具栏提供快捷操作按钮及功能菜单，工具箱提供常用的 Web 控件，用于用户的页面设计，代码视图编辑区用于编辑界面及后台代码，错误调试显示窗口用于显示程序运行过程中的错误信息，解决方案管理区用于网站项目文件管理及控件的属性窗口切换操作。

能力大比拼，看谁做得又好又快

使用 TextBox 控件并显示“ 欢迎使用 VS2012 ”，结果如图 1-8 所示。

图 1-8　程序运行效果

学习小结

——你掌握了吗？

1. 使用 Visual Studio 2012 创建 Web 应用程序
2. 掌握页面载入事件的作用

1.2　ASP.NET 程序构成及页面事件

学习要点

1. 掌握 ASP.NET 页面结构选项

2．掌握 ASP.NET 应用程序文件夹
3．掌握 ASP.NET 的页面指令
4．掌握 ASP.NET 的页面事件

导学实践，跟我学

【案例】ASP.NET 程序构成初探。

在创建 ASP.NET Web 应用程序时，不可避免地要学会并掌握页面结构、指令、事件，以及应用程序文件夹、Global.asax 及程序的编译。下面继续以“ Hello VS2012 为例讲解以上的知识点。

具体步骤如下。

（1）打开 Default.aspx 页面并单击“ 源 ”，如图 1-9 所示。

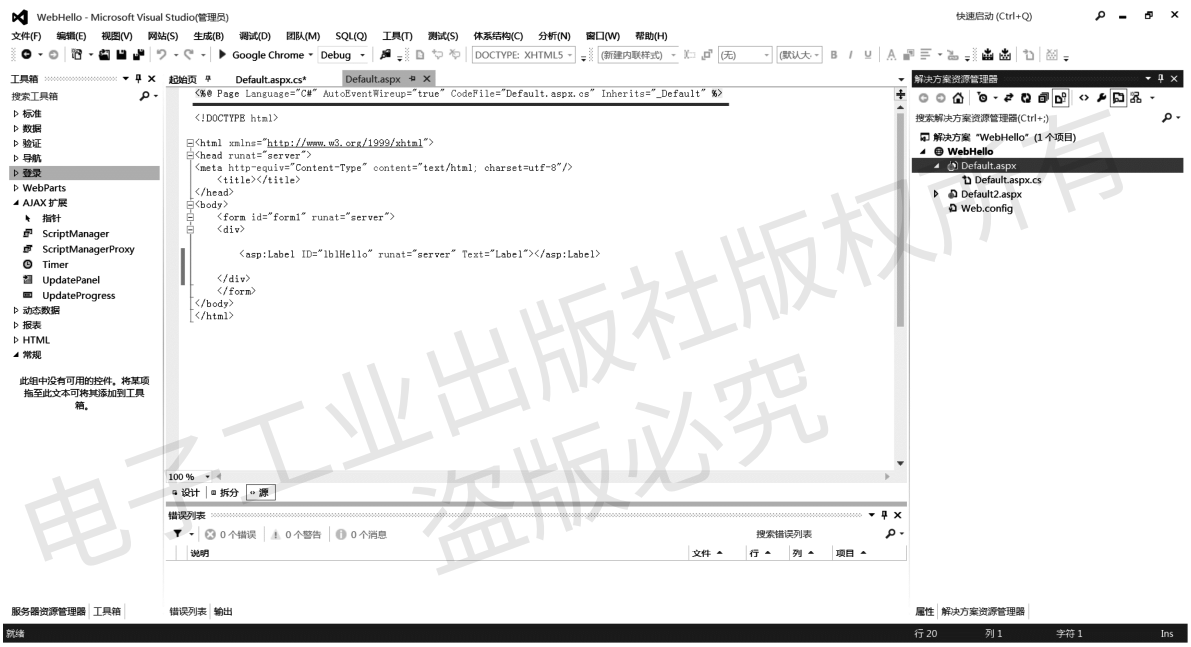

图 1-9　Default.aspx 源

ASP.NET 指令在每个 ASP.NET 页面中都有。使用这些指令可以控制 ASP.NET 页面的行为。上面是 Page 指令的一个例子。@Page 指令允许为 ASP.NET 页面（.aspx）指定解析和编译页面时使用的属性和值，这是最常用的指令。ASP.NET 页面是 ASP.NET 的一个重要部分，所以它有许多属性。详见背景知识。

（2）切换到 Default.aspx.cs，如图 1-10 所示。

Page_Load()事件是页面事件中最常用的一个，它表示页面载入时触发的事件，一般情况下用来给窗体控件初始化值。

（3）创建“ Global.asax ”。在“ 解决方案资源管理器 ”的网站根目录上右击选择“ 添加新项 ”选项，弹出如图 1-11 所示的对话框，在“ 添加新项 ”对话框中执行“ 全局应用程序类 ”→“ 添加 ”命令，再在“ 解决方案资源管理器 ”中双击 Global.asax，如图 1-12 所示。

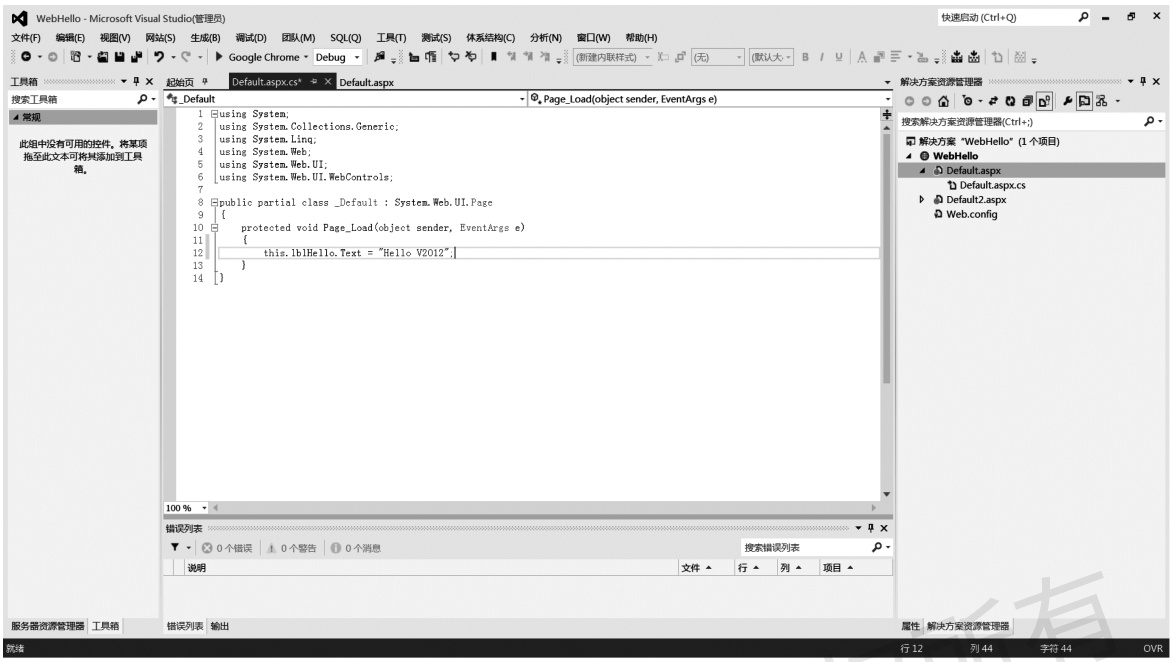

图 1-10　Default.aspx.cs 代码视图

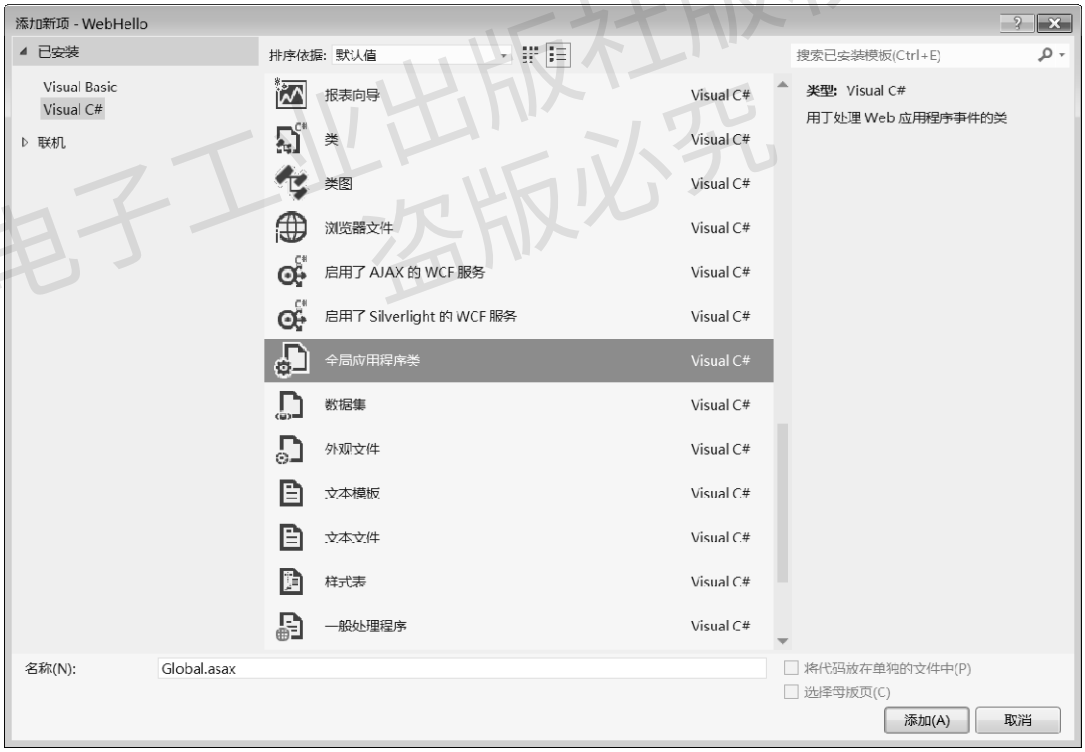

图 1-11　创建 Global.asax

ASP.NET 应用程序只能有一个 Global.asax 文件，它自动生成的事件常用的主要有 Application_Start()、Application_End()、Session_Start()、Session_End()等。

```
<%@ Application Language="C#" %>

<script runat="server">

    void Application_Start(object sender, EventArgs e)
    {
        // 在应用程序启动时运行的代码

    }

    void Application_End(object sender, EventArgs e)
    {
        //  在应用程序关闭时运行的代码

    }

    void Application_Error(object sender, EventArgs e)
    {
        // 在出现未处理的错误时运行的代码

    }

    void Session_Start(object sender, EventArgs e)
    {
        // 在新会话启动时运行的代码

    }

    void Session_End(object sender, EventArgs e)
    {
        // 在会话结束时运行的代码。
        // 注意: 只有在 Web.config 文件中的 sessionstate 模式设置为
        // InProc 时，才会引发 Session_End 事件。如果会话模式设置为 StateServer
        // 或 SQLServer，则不引发该事件。

    }

</script>
```

100 %

图 1-12　Global.asax 代码

背景知识

1．页面指令

ASP.NET 指令在每个 ASP.NET 页面中都有。使用这些指令可以控制 ASP.NET 页面的行为。下面是 Page 指令的一个例子。

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs" Inherits="_Default" %>
```

在 ASP.NET 页面或用户控件中有 11 个指令。无论页面是使用后台编码模型还是内置编码模型，都可以在应用程序中使用这些指令。

基本上，这些指令都是编译器编译页面时使用的命令。把指令合并到页面中是很简单的。指令的格式如下：

```
<%@ [Directive] [Attribute=Value] %>
```

在上面的代码行中，指令以“ <%@ ”开头，以“ %> ”结束。最好把这些指令放在页面或控件的顶部，因为开发人员传统上都把指令放在那里(但如果指令位于其他地方，页面仍能编译)。当然，也可以把多个属性添加到指令语句中，如下所示：

```
<%@ [Directive] [Attribute=Value] [Attribute=Value] %>
```

表 1-1 描述了 ASP.NET 页面中的常用指令。

表 1-1　ASP.NET 页面中常用指令及说明

指　令	说　明
Assembly	把程序集链接到与它相关的页面或用户控件上
Control	用户控件（.ascx）使用的指令，其含义与 Page 指令相当

续表

指　令	说　明
Implements	实现指定的.NET Framework 接口
Import	在页面或用户控件中导入指定的命名空间
Master	允许指定 master 页面—— 在解析或编译页面时使用的特定属性和值。这个指令只能与 master 页面（.master）一起使用
MasterType	把类名与页面关联起来，获得包含在特定 master 页面中的强类型化的引用或成员
OutputCache	控制页面或用户控件的输出高速缓存策略
Page	允许指定在解析或编译页面时使用的页面特定属性和值。这个指令只能与 ASP.NET 页面（.aspx）一起使用
PreviousPageType	允许 ASP.NET 页面处理应用程序中另一个页面的回送信息
Reference	把页面或用户控件链接到当前的页面或用户控件上
Register	给命名空间和类名关联上别名，作为定制服务器控件语法中的记号

1）@Page

@Page 指令允许为 ASP.NET 页面（.aspx）指定解析和编译页面时使用的属性和值。

2）@Master

@Master 指令类似于@Page 指令，但@Master 指令用于 master 页面（.master）。在使用@Master 指令时，要指定和站点上的内容页面一起使用的模板页面的属性。内容页面（使用@Page 指令建立）可以继承 master 页面上的所有 master 内容（在 master 页面上使用@Master 指令定义的内容）。尽管这两个指令是类似的，但@Master 指令的属性比@Page 指令少。

3）@Control

@Control 指令类似于@Page 指令，但@Control 指令是在建立 ASP.NET 用户控件时使用的。@Control 指令允许定义用户控件要继承的属性。这些属性值会在解析和编译页面时赋予用户控件。@Control 指令的可用属性比@Page 指令少，但其中有许多都可以在建立用户控件时进行的修改。

4）@Import

@Import 指令允许指定要导入到 ASP.NET 页面或用户控件中的命名空间。导入了命名空间后，该命名空间中的所有类和接口就可以在页面和用户控件中使用了。这个指令只支持一个属性 Namespace。

Namespace 属性带一个 String 值，它指定要导入的命名空间。@Import 指令不能包含多个属性/值对。所以，必须把多个命名空间导入指令放在多行代码上，如下所示：

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
```

5）@Implements

@Implements 指令允许 ASP.NET 页面实现特定的.NET Framework 接口。这个指令只支持一个 Interface 属性。

Interface 属性直接指定了.NET Framework 接口。ASP.NET 页面或用户控件实现一个接口时，就可以直接访问其中的所有事件、方法和属性。

6）@Register

@Register 指令把别名与命名空间和类名关联起来，作为定制服务器控件语法中的记号。把一个用户控件拖放到.aspx 页面上时，就使用了@Register 指令。把用户控件拖放到.aspx 页面上，Visual Studio 2012 就会在页面的顶部创建一个@Register 指令。这样就在页面上注册了用户控件，该控件

就可以通过特定的名称在.aspx 页面上访问了。

7）@Assembly

@Assembly 指令在编译时把程序集（.NET 应用程序的构建块）关联到 ASP.NET 页面或用户控件上，使该程序集中的所有类和接口都可用于页面。这个指令支持 Name 和 Src 两个属性。

（1）Name：允许指定用于关联页面文件的程序集名称。程序集名称应只包含文件名，不包含文件的扩展名。例如，如果文件是 MyAssembly.vb，Name 属性值应是 MyAssembly。

（2）Src：允许指定编译时使用的程序集文件源。

8）@PreviousPageType

这个指令用于指定跨页面的传送过程起始于哪个页面。

@PreviousPageType 指令是一个新指令，用于处理 ASP.NET 2.0 提供的跨页面传送新功能。这个简单的指令只包含 TypeName 和 VirtualPath 两个属性。

（1）TypeName：设置回送时的派生类名。

（2）VirtualPath：设置回送时所传送页面的地址。

9）@MasterType

@MasterType 指令把一个类名关联到 ASP.NET 页面上，以获得特定 master 页面中包含的强类型化引用或成员。这个指令支持以下两个属性。

（1）TypeName：设置从中获得强类型化的引用或成员的派生类名。

（2）VirtualPath：设置从中检索这些强类型化的引用或成员的页面地址。

下面是它的一个例子。

```
<%@ MasterType VirtualPath="~/Wrox.master" %>
```

10）@OutputCache

@OutputCache 指令控制 ASP.NET 页面或用户控件的输出高速缓存策略。这个指令支持 10 个属性。

11）@Reference

@Reference 指令声明，另一个 ASP.NET 页面或用户控件应与当前活动的页面或控件一起编译。这个指令支持以下两个属性。

TypeName：设置从中引用活动页面的派生类名。

VirtualPath：设置从中引用活动页面的页面或用户控件地址。

下面是使用@Reference 指令的一个例子。

```
<%@ Reference VirtualPath="~/MyControl.ascx" %>
```

2．页面事件

在 ASP.NET 页面的寿命周期内，Page 对象会公开一些被频繁使用的标准事件。ASP.NET 页面框架在运行时，会自动调用这些方法的相应代理实例。这样用户就无须编写必要的“ 粘接代码 ”。以下是按启动顺序提供运行时连续的代理实例。

（1）Page_Init：出现此事件期间，用户可以初始化值或连接可能具有的任何事件处理程序。

（2）Page_Load：出现此事件期间，用户可以执行一系列的操作来首次创建 ASP.NET 页面或响应由投递引起的客户端事件。在此事件之前，已还原页面和控件视图状态。使用 IsPostBack 页面属性检查是否为首次处理该页面。如果是首次处理，应执行数据绑定。此外，读取并更新控件属性。

（3）Page_DataBind：在页面级别调用 DataBind 方法时，将引发 DataBind 事件。如果在单个控件上调用 DataBind，则它只激发它下面控件的 DataBind 事件。

（4）Page_PreRender：恰好在保存视图状态和呈现控件之前激发 PreRender 事件。用户可以使用此事件在控件上执行所有最后时刻的操作。

（5）Page_Unload：完成页面呈现之后，将激发 Page_Unload 事件。此事件执行最终清理工作，包括清理打开的数据库连接、丢弃对象或关闭打开的文件等。

以下概括了非确定性事件。

（1）Page_Error：如果在页面处理过程中出现未处理的程序，则激发 Error 事件。错误事件为用户提供了妥善处理错误的机会。

（2）Page_AbortTransaction：如果要指明交易是成功还是失败，交易事件非常有用。此事件通常用于购物车方案，其中此事件可以指示订购是成功还是失败。如果已终止交易，则激发此事件。

（3）Page_CommitTransaction：如果已成功提交交易，则激发此事件。

除了上面的页面事件之外，还有以下事件。

（1）InitComplete：表示页面完成了初始化。

（2）LoadComplete：表示页面完全加载到内存中。

（3）PreInit：表示页面初始化前的那一刻。

（4）PreLoad：表示页面加载到内存前的那一刻。

（5）PreRenderComplete：表示页面显示在浏览器中之前的那一刻。

这些新页面事件的构建与前面介绍的页面事件相同。

如果创建一个 ASP.NET 页面，并打开跟踪功能，就会看到页面事件的启动顺序，它们按照下面的顺序启动。

（1）PreInit

（2）Init

（3）InitComplete

（4）PreLoad

（5）Load

（6）LoadComplete

（7）PreRender

（8）PreRenderComplete

（9）Unload

添加了这些新的页面事件后，就可以在页面编译期间在许多不同的地方处理页面和页面上的控件。

3．ASP.NET 应用程序文件夹

1）\App_Code 文件夹

App_Code 文件夹是存储 classes、.wsdl 文件和 typed datasets 的地方。用户的解决方案中的所有页面可以自动访问存储在这个文件夹的任何一个项目。如果这些项目是一个 class（.vb 或.cs），则 Visual Studio 2012 会自动检测并编译它；也会自动地创建源于.wsdl 文件的 XML Web service proxy class；或者一个源于.xsd 文件的一个 typed dataset。

2）\App_Data 文件夹

App_Data 文件夹是应用程序存储数据的地方，可以包括 Microsoft SQL Express 文件（.mdf files）、Microsoft Access 文件（.mdb files）、XML 文件等。

3）\App_Themes 文件夹

App_Themes 文件夹是存储 ASP.NET 2.0 新特性主题需要使用的.skin 文件, CSS 文件和 images 文件的地方。

4）\App_GlobalResources 文件夹

资源文件（.resx）是一个在用户的应用程序中依据不同文化来改变页面内容的可以作为数据字典的字串表。除字串外，还可添加 image 等其他文件。

5）\App_LocalResources 文件夹

用户也可以把资源文件添加到\App_LocalResources 文件夹，只不过\App_GlobalResources 文件夹是应用程序级别，而\App_LocalResources 文件夹是页面级别。

6）\App_WebReferences 文件夹

用户可以使用\App_WebReferences 文件夹自动访问在自己的应用程序中引用的远程 Web Services。

7）\App_Browsers 文件夹

存储在\App_Browsers 文件夹中的.browser 文件，用户也可以用它来判断浏览器的兼容性。

4．Global.asax

ASP.NET 应用程序只能有一个 Global.asax 文件，该文件支持许多项。创建它后，就会得到如下模板。

与处理.aspx 页面中页面级的事件一样，也可以在 Global.asax 文件中处理应用程序的事件。除了这个代码示例中列出的事件之外，还可以在这个文件中构建以下事件。

（1）Application_Start：在应用程序接收到第一个请求时调用，这是在应用程序中给应用程序级的变量赋值或指定对所有用户必须保持的状态的理想位置。

（2）Session_Start：类似于 Application_Start 事件，但这个事件在用户第一次访问应用程序时调用。例如，Application_Start 事件只在接收到第一个请求时触发，第一个请求会让应用程序运行，而 Session_Start 事件会在每个终端用户第一次向应用程序发出请求时调用。

（3）Application_BeginRequest：该事件会在每个请求发出之前触发。也就是说，在请求到达服务器，且得到处理之前，会触发 Application_BeginRequest 事件，并在处理该请求之前处理。

（4）Application_AuthenticateRequest：每个请求都会触发该事件，允许为请求建立定制的身份验证。

（5）Application_Error：在应用程序的用户抛出一个错误时触发。它适合于提供应用程序级的错误处理，或者把错误记录到服务器的事件日志中。

（6）Session_End：在 InProc 模式下运行时，这个事件在终端用户退出应用程序时触发。

（7）Application_End：在应用程序结束时触发。大多数 ASP.NET 开发人员都不使用这个事件，因为 ASP.NET 很好地完成了关闭和清理剩余对象的任务。

学习小结

——你掌握了吗？

1．ASP.NET 2.0 的页面指令

2．ASP.NET 2.0 的应用程序文件夹

3．ASP.NET 2.0 的页面事件

4．Global.asax 文件中的事件